

TR 00.1318
August 30, 1965

Technical Report

The Design of Processor Controls Using
a Read-Only Storage

Dorothy L. Schnabel

The Design of Processor Controls Using a Read-Only Storage

by

Dorothy L. Schnabel

ABSTRACT

This paper describes some of the considerations made in the design of the control of a processor using a read-only storage.

Using an IBM System/360 Model 50 as an example, the paper shows how a read-only storage (ROS) word is used to control the processor action in one state and provide sequence information for determining the next state.

The Model 50 ROS word is divided into four segments, each having the following chief function.

- Segment one--supplies the data flow control information for action in the present state.
- Segment two--supplies control information for remembering significant states and state events which will determine future action.
- Segment three-- supplies a constant for use as control or data information.
- Segment four--provides information for determining the address of the next control word to be selected from the ROS.

Emphasis is placed on the design of the control word structure and its decoding rather than on microprogramming techniques used in the execution of program instructions.

IBM 0360-50
Storage
Microprogramming
Read-Only
07 Computers

IBM
®

International Business Machines Corporation
Systems Development Division, Poughkeepsie, New York

CONTENTS

IBM System/360 Model 50 CPU	1
The Design of the Model 50 Processor Controls	2
Read-Only Storage Word	3
Specification of Data Flow Controls	6
Increasing the Bit Efficiency of the ROS word	8
Status Control Field	8
Emit Field	9
Read-Only Storage Word Addressing	10
Specification of Branch Test Micro-Orders	12
Conclusions	14

IBM SYSTEM/360 MODEL 50 CPU

As shown in Figure 1, there are two parallel data paths in the Model 50. One is a 32-bit wide binary adder and shifter path. The L register feeds into the left side of the adder (X) via a true or complement gate, (T/C) and the R, M, and H registers feed into the right side of the adder (Y). The L, R, M, and H registers are also 32 bits wide.

The adder output can be shifted either right or left by one or four bits. The shifter output latch (T) goes to a bus feeding the L, R, M, and H registers and also to the main storage address register, SAR, and the main storage data register, SDR.

The Model 50 has a maximum 256K byte main storage, also 32 bits wide, which operates on a 2.0- μ sec. cycle. The storage is addressed from a 24-bit storage address register. Data is transferred to and from storage via the storage data register.

A 24-bit instruction address register and counter feeds into SAR.

The other parallel data path is 8 bits wide and is called the mover. It performs logical operations such as the AND, OR, and EXCLUSIVE OR of 8-bit data groups

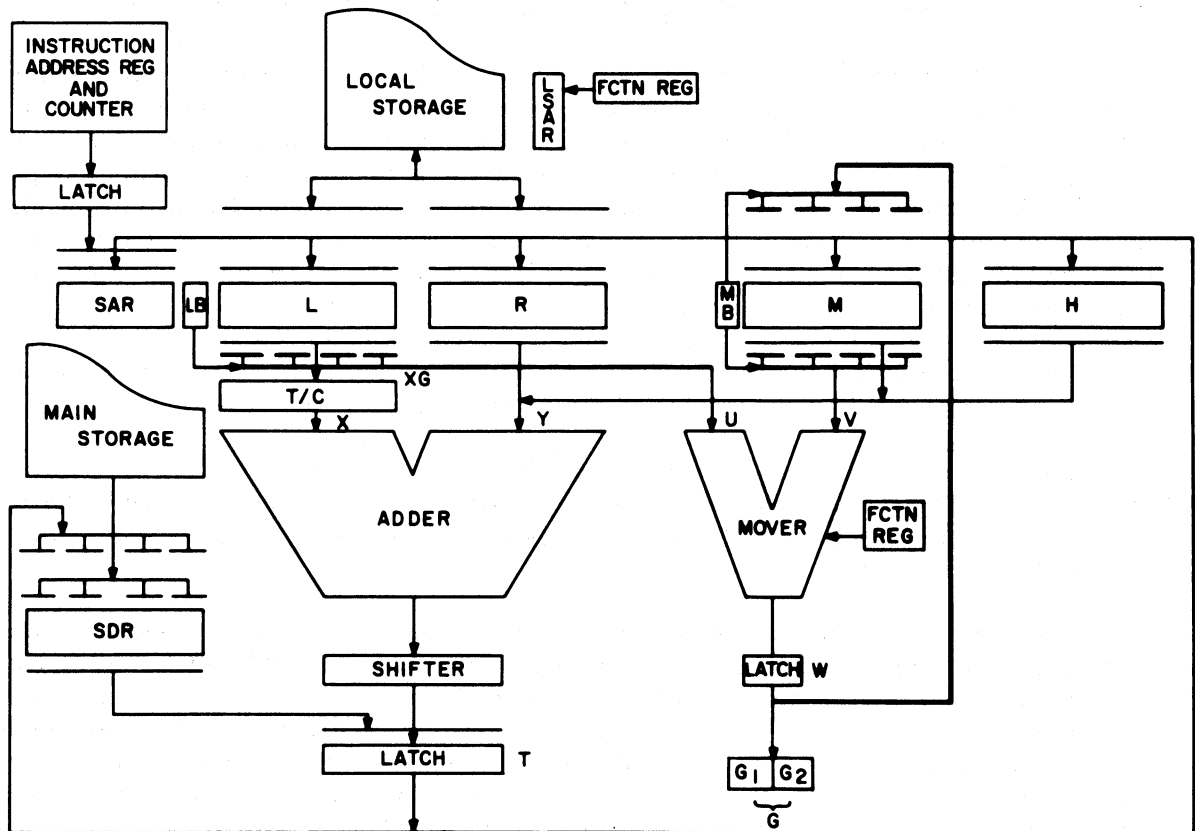


Figure 1. IBM System/360 Model 50 simplified data flow.

called bytes, and also can merge half bytes from two sources. The left mover input byte is selected from the L register under control of a 2-bit counter called LB. The right mover input byte is selected from the M register under control of another 2-bit counter called MB. The left and right inputs are referred to as (U) and (V) respectively. The mover output byte goes to a latch (W) and may be inserted into a byte of the M register under control of the MB counter.

The mover is also used to set two 4-bit operand field length counters G1 and G2. G1 and G2 hold the field lengths of operands one and two respectively in the execution of decimal instructions. For the execution of variable field length character type instructions, which can have a maximum field length of 256 bytes, counters G1 and G2 are coupled to act as an 8-bit "G" counter.

The Model 50 has a local storage of sixty-four 32-bit words and is addressed by a 6-bit address register, LSAR. The local storage is divided into four quadrants. One contains the 16 general-purpose registers and another the four 64-bit floating-point registers which are part of the System/360 structure.¹ A third quadrant contains 16 registers which are used as a scratch pad storage in the execution of CPU instructions. The fourth quadrant is similarly used for channel controls.

The local storage operates on a 0.5- μ sec cycle which is the same as the basic CPU cycle. This is the time required for a register to register transfer.

THE DESIGN OF THE MODEL 50 PROCESSOR CONTROLS

The IBM System/360 generation of machines has brought the term, "micro-programming,"^{1,2} into common usage in the field of computer design. In the past generation of machine, the specification of the action of a machine in one cycle or state, and the sequencing to the next state during a program instruction execution was implemented with logical AND and OR circuits. The specification was simply called sequential control design. Microprogramming is this same control design but the implementation is different and implies the use of a read-only storage which contains state and sequence controls in a program-like form.

One read-only storage word, called a microinstruction, controls a machine cycle. A sequence of microinstructions is called a microprogram.

The essence of processor control design is: 1) to specify for every machine cycle the exact action of the machine, i. e., the register gates that are to be opened and closed, the carry control, the selection of main storage, etc., and 2) to specify the next machine cycle.

The objective is the same regardless of whether the control design is implemented with hardware logical blocks or with a read-only storage. ROS characteristics, however, cause design considerations to be different from those made in a hardware-controlled machine. The ROS word has a maximum realizable width. All control information for the data flow is confined to these bits. Branching conditions tend to be more general and sometimes impose timing limitations. The control ability can be expanded by increasing the amount of hardware decoding of the ROS data register output; but to an extreme, this defeats the purpose of having a read-only storage.

The controls in a ROS machine tend to be more formalized than those in a hardware machine. This proves to be a great advantage in the simulation of the control design before implementation.

The fact that the control specification is largely in a storage device makes design changes much easier. A storage unit plane can be replaced without any hardware changes.

Once the gating controls have been defined for a data flow, all microprogrammers are free to activate this gating. Let us assume that ten engineers are assigned to design a hardware-controlled machine and that each wishes to activate a particular gate, say an adder input, under 20 different conditions. A two-hundred way logical OR could be used to gather all conditions, or the ten designers could integrate their logic, making debugging, ease of understanding, and change very difficult.

This problem does not exist in a microprogram-controlled machine. A gate may be activated by each word in the ROS and conceivably can come from thousands of conditions.

It is the purpose of this paper to show some of the general considerations made in the design of a ROS-controlled machine and yet cite specific examples from the design of the System/360 Model 50 computer.

READ-ONLY STORAGE WORD

The Model 50 ROS word has four control segments as shown in Figure 2.

The first segment contains encoded data which will control all data flow action such as the gating to and from registers, the insertion of a carry into an adder, counter control, and the selection of main and local storage.

The second segment will be called status control and has two purposes:

1. The decoded control information is used to interrogate data conditions such as an adder carry, an adder sum, or a sign value, and to set a trigger called a stat to remember the results of the interrogation for future control action.
2. The status control information is also used to modify the controls specified by the data flow control segment of the ROS word. For example, the status control field is used to set a function register which will modify a data flow control field to cause data to be logically ANDed or ORed or EXCLUSIVE ORed depending on the value set into the function register. Figure 1 shows the mover function register.

The third control segment in the ROS word, called the emit field, is used as the source of a constant. It is used in conjunction with the data flow control segment for sign and zone insertion, as an adder input, and also for addressing the local storage.

The emit field provides a constant to the status control segment to set a function register and also to set a stat unconditionally.

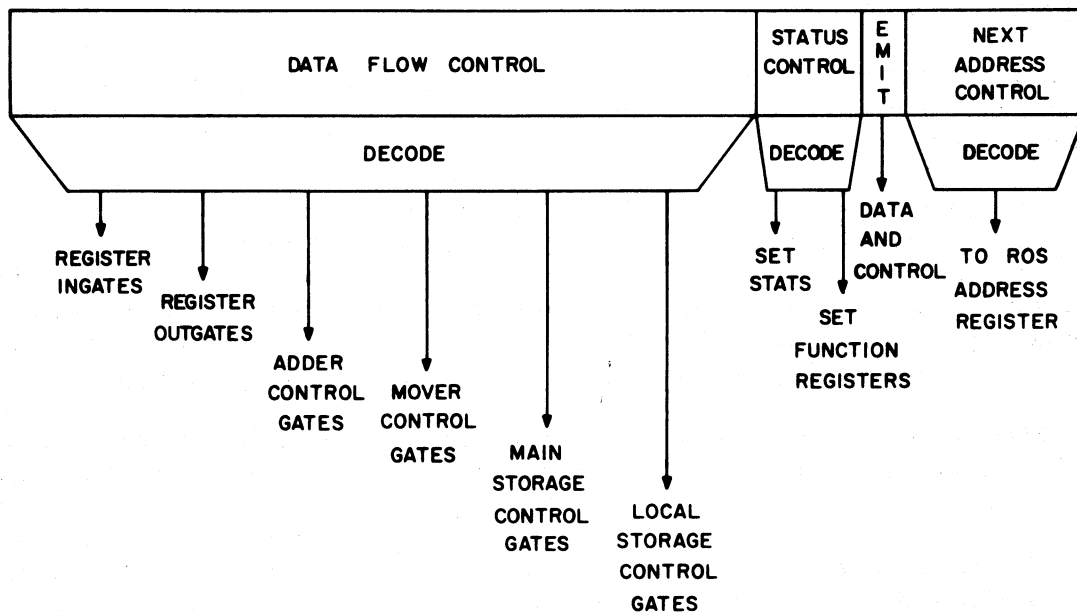


Figure 2. Read-only storage word.

Both the data flow control and status control segments may use the emit field simultaneously.

The fourth ROS word segment is used to determine the address of the next read-only storage word to be fetched for control of the next machine state. This field contains a base address, which forms the high-order portion of the ROS word address, and control information to interrogate the status or data and parameter conditions to form the low-order address bits. These low-order bits are called branch bits.

The complete ROS word address formation is detailed in the "Read-Only Storage Word Addressing" section.

The basic CPU on the Model 50 has a ROS word width of 90 bits allocated as follows:

data flow control	50 bits
status control	6
emit field	4
address control	24
ROS word parity	3
spare bits	3
	90

See Figures 3 and 4 for details on the ROS word controls and data flow gating relationship, and the timing of each on the Model 50.

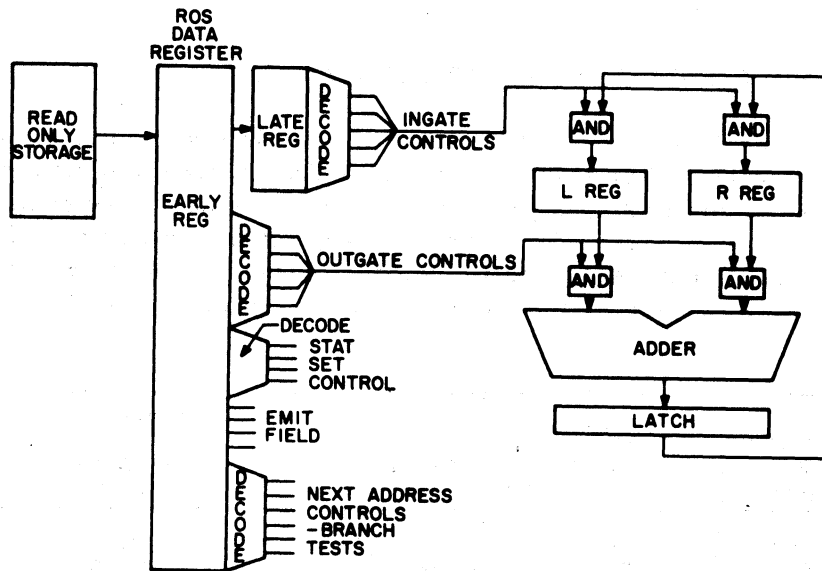


Figure 3. Relationships between ROS control word and data flow.

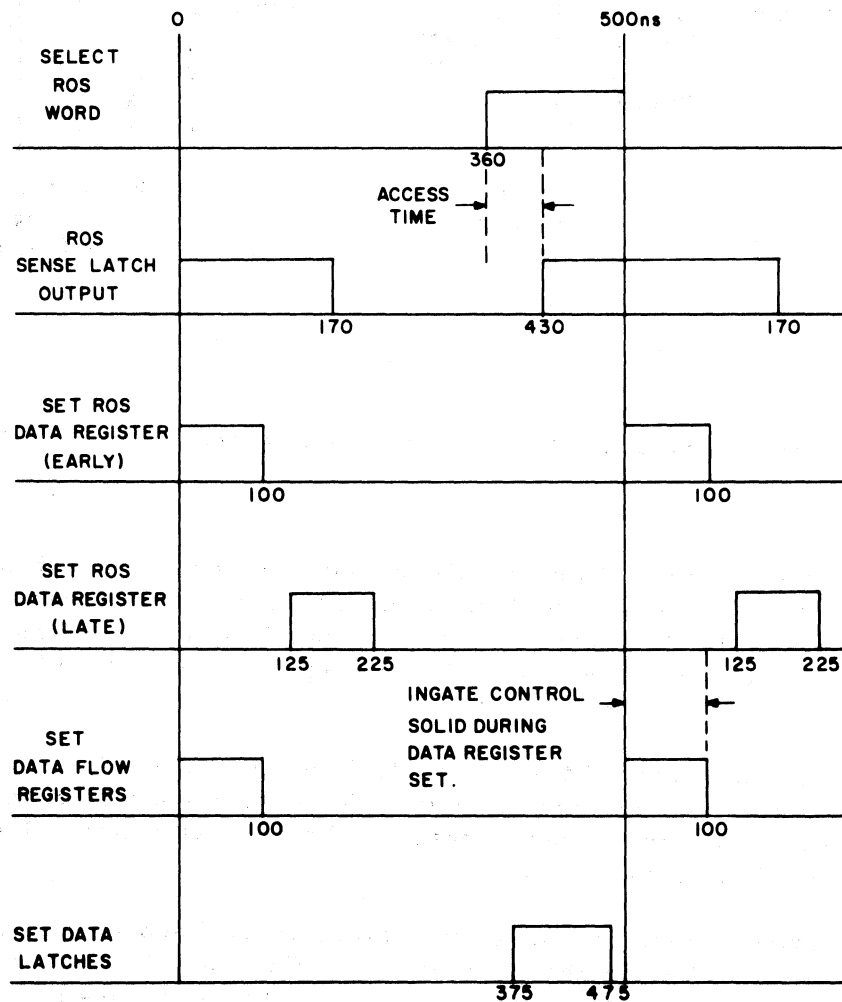


Figure 4. IBM System/360 Model 50 ROS and data flow timing.

SPECIFICATION OF DATA FLOW CONTROLS

As previously mentioned, the ROS word is divided into segments containing coded control information. For instance, the data flow control segment contains numerous fields, each controlling a particular area of the data flow such as register gating to the adder, gating of the adder output to registers, selection of local storage, etc. Each distinct code in a field is called a micro-order. Figure 5 shows some of the control fields and associated micro-orders for the data flow shown in Figure 1.

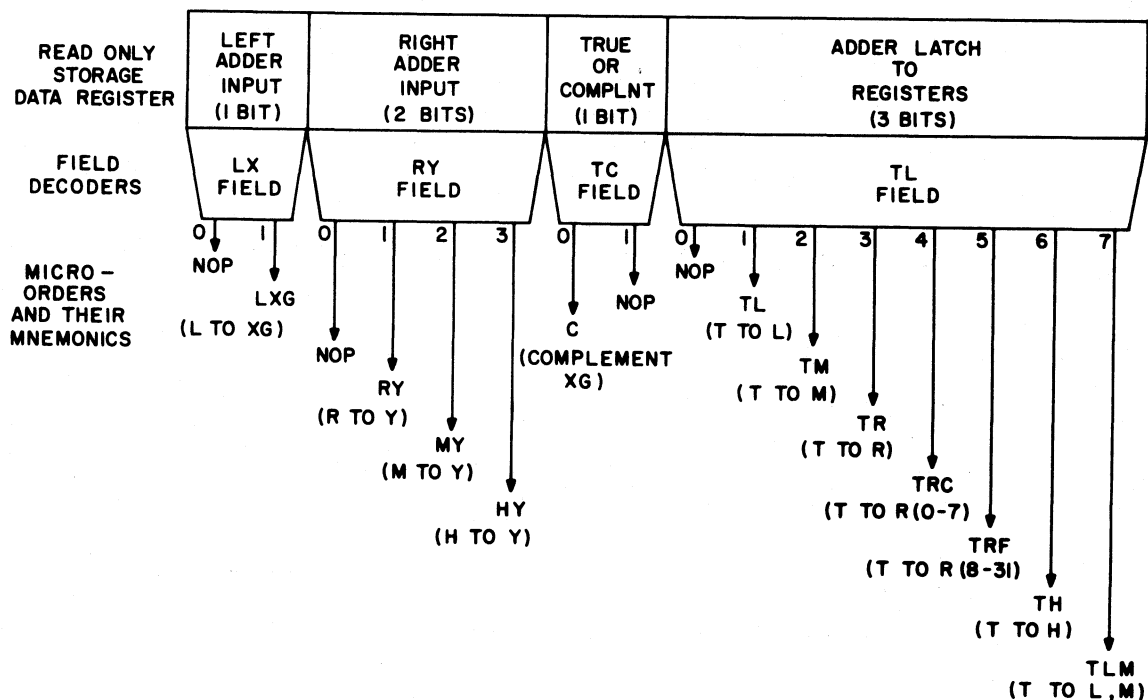


Figure 5. Simplified adder fields in data flow control segment of word.

There are over 30 control fields (this includes fields used in I/O mode which are mutually exclusive with the fields used in CPU mode as explained in "Increasing the Bit Efficiency of the ROS Word") and more than 200 micro-orders in the data flow control segment of the Model 50. The average field contains 3 bits and therefore a maximum of eight micro-orders.

A very important aspect of the ROS control design is the definition of the micro-orders and their grouping into control fields. This micro-order specification establishes the microprogramming language of the processor from which all microprograms will be constructed.

The microprogrammer designing the original control sequencing of the CPU has the advantage of taking part in the micro-order definition. This early communication between the microprogram control designer and the hardware designer is absolutely essential. Subsequent microprogrammers who write microprograms for the emulation

of another machine or who microprogram special features, however, must work within the established framework and may make only minor modifications and additions to the control specifications. This is not so in a hardware-controlled machine.

It is therefore the burden of the original microprogrammer to specify the control fields and their micro-orders to best satisfy the performance objectives of the basic CPU while keeping the controls general enough to accommodate the future microprogramming of emulators and special features.

It is not practical to attempt to completely specify the ROS word format until the control designer has sequenced almost completely the execution of each program instruction and the instruction fetch and decode routine.

As a first pass at the control word specification, the basic data flow should be examined for independent data handling areas. On the Model 50 this is the adder path, the mover path, main storage and local storage. Each one of these areas is assigned several control fields for in-gating, out-gating and data manipulation. For example, the Model 50 adder path has the following control fields:

1. left adder input
2. right adder input
3. true or complement of left input
4. carry detection
5. shifter
6. adder latch to register

Next, micro-orders within a field are specified for those gates that are mutually exclusive and for combinations of gates which occur most frequently. See Figure 5. Since only one source may feed data into the right side of the adder at a time, the micro-orders controlling gates from R, M, and H are all mutually exclusive and placed in one control field.

As previously mentioned, a micro-order may activate more than one gate. For example, a Model 50 control field is used to decrement length counters G_1 and G_2 . The following micro-orders are in the same field:

- DG_1 , which decrements only counter G_1 ;
- DG_2 , which decrements only counter G_2 ; and
- DG_1G_2 , which decrements both counters G_1 and G_2 .

It is impossible to specify a micro-order for each gating combination likely to occur and still have a realizable ROS word width. When assigning a micro-order to control multiple gates, judgment must be used as to the use frequency of the combination and the time that will be sacrificed by doing the same job serially with successive microinstructions using simple micro-orders.

INCREASING THE BIT EFFICIENCY OF THE ROS WORD

The micro-orders controlling the data flow require the largest segment of the ROS word. The bit efficiency of a control field can be increased by increasing the field width and assigning more micro-orders to a field. On the Model 50, some totally unrelated fields were combined into one; yet all micro-orders remained mutually exclusive. An example is the combination of a field used in decimal instruction execution to test for invalid decimal digits with a field used in instruction preparation to increment the instruction counter. See Figure 6.

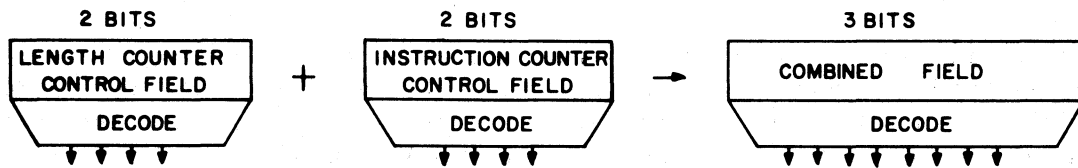


Figure 6. Example showing increasing bit efficiency of ROS word.

A limiting factor in extending field width, besides mutual exclusion, is the amount of hardware and number of logical levels permissible in decoding the field.

Another technique used is the dual decoding of ROS bits. On the Model 50, the I/O channels use the CPU to update addresses and counts and to store data in main storage. These operations are performed in I/O mode - which is mutually exclusive with CPU mode. Those ROS bits which control an area of the data flow used only in CPU mode may be used to control or provide test information for the I/O channels in I/O mode.

For example, the field length and byte counter controls are not used in I/O mode. These control bits are coded to control gating and stat setting functions for the multiplex channel in I/O mode. Since the channels do not use any complicated adder functions such as those needed in floating-point and decimal arithmetic, this adder control field is used to provide control information to the selector channels in I/O mode.

STATUS CONTROL FIELD

As mentioned in the section on "Read-Only Storage Word," one function of this field is to set stats to remember conditions for future microprogram branching. The Model 50 has eight general-purpose (GP) stats which may be set as the result of specific tests of conditions such as a zero adder sum or a decimal carry. A GP stat may be set unconditionally to a value specified by a bit in the emit field. An example of this usage is found in the microprogram for floating-point multiply and divide. A general-purpose stat is turned on in the first microinstruction selected for the divide instruction execution following instruction preparation. The first microinstruction for the multiply instruction turns off this same general-purpose stat. A common microprogram is then entered for the prenormalization of operands. Just prior to exponent arithmetic and inner loop execution, a branch is taken on the GP stat set at the end of instruction decoding to the routine for multiply or divide.

There are also several sign stats which hold the value of a fixed, floating-point, or decimal sign. In addition, byte stats are used to control the transfer of data from the CPU to main storage on a byte basis. See Figure 1.

An important function of the status control field is to set a function register for control of the data flow. The Model 50 has two function registers. One is a two-bit register which addresses one of the four local storage sectors for general-purpose, floating-point, working, or I/O registers.

The other function register is three bits wide, and controls data manipulation in the mover path. Six System/360 VFL instructions — move, move numeric, move zone, logical OR, logical AND, and logical EXCLUSIVE OR — are executed by one microprogram. The objective in each one of these instructions is to form a byte which is the function of two source bytes. If the source bytes are in the L and M registers (See Figure 1) and the resultant byte M_B^1 is to be stored into M, then

$$M_B^1 = f(L_B, M_B)$$

One microprogram selects successive byte pairs, stores the resultant byte in the CPU and fetches more data from main storage as needed. The function $f(L_B, M_B)$ is controlled by the function register which is set differently for each of the six instructions at the end of instruction preparation. Only one micro-order is needed to activate the function register for data manipulation. This technique has resulted in the saving of many microinstructions and five of the six micro-orders needed to directly specify the data function for these six System/360 instructions.

Another function of the status control field is the use of the emit field to inject a constant into the data flow as explained in the next section.

EMIT FIELD

Because the four-bit emit field on the Model 50 serves as a constant, it is the only ROS word segment which is not decoded. The emit field is used to:

1. set stats directly to the emit bit values, or on or off according to the one-bits present;
2. address a register in a sector of local storage;
3. enter a constant into the adder path;
4. enter a constant into the mover path.

The clever microprogrammer can choose working storage register addresses and general-purpose stats so that one emit value can satisfy several or all of the above functions in the same microinstruction.

READ-ONLY STORAGE WORD ADDRESSING

As mentioned in the section, "Read-Only Storage Word," one segment of the ROS word contains information for determining the address of the next ROS word to be fetched. If the next state were always unconditionally one particular state, the present ROS word could contain the complete address of the next word, or the next selected word could be one having the next higher address. However, it is very frequently desirable to go to one of several states depending on the presence of a condition occurring in the present state, the last state, some earlier state, or a combination of these.

This branching to one of several states is accomplished by dividing the next word address into two segments. The high-order segment is a constant and is contained in the present ROS word. The low-order, or branching segment, is determined by testing machine conditions, such as an adder carry, counter value, or stats set in a previous cycle. The present ROS word must contain control information which when decoded will determine the machine conditions and stats to be tested.

On the Model 50, twelve bits are used to address a 2816-word ROS. The high-order six bits, called the base address, are contained in the present ROS word and designate 64-word blocks of storage. The next four bits may be obtained directly from the present ROS word or from data or parameter conditions in the CPU. This latter technique is known as function branching. The low two-bits are called A and B branch bits and are determined by the testing of data and parameter values, and stat settings.

As an example, assume that a particular microinstruction (ROS word) is to be followed by one of two microinstructions depending upon the presence of an adder carry occurring in the present cycle. The microinstruction specifies the high-order eleven bits of the next microinstruction address; the low-order bit is determined by the presence or absence of the carry. Figure 7 shows how the ROS address is formed.

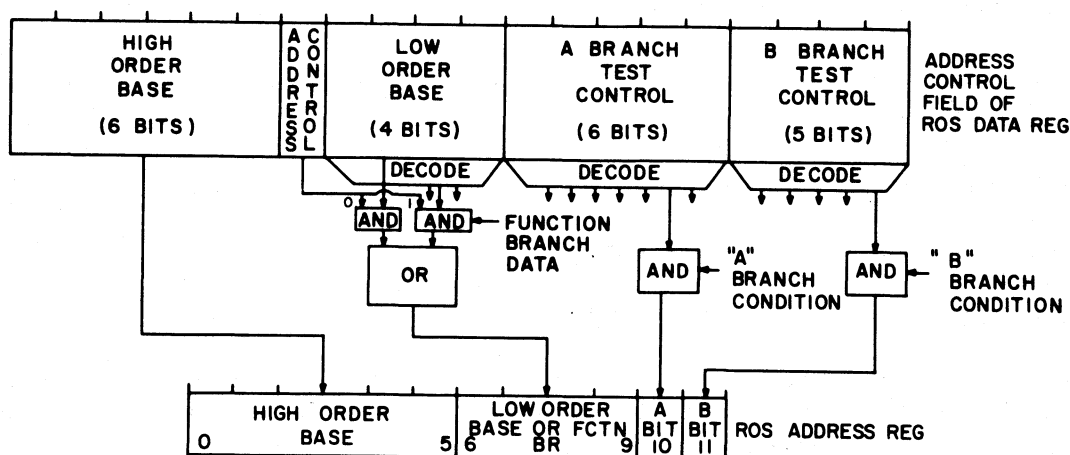


Figure 7. Formation of address of next ROS word.

An example of a microprogram is given in Figure 8. The microinstruction at ROS location 453 contains a branch micro-order BCARY which tests the carry out of the adder. BCARY is called an immediate branch because it tests a condition which is the result of action occurring in the present cycle. The Model 50 has 32 B-branch test micro-orders, but only two are immediate because of the time

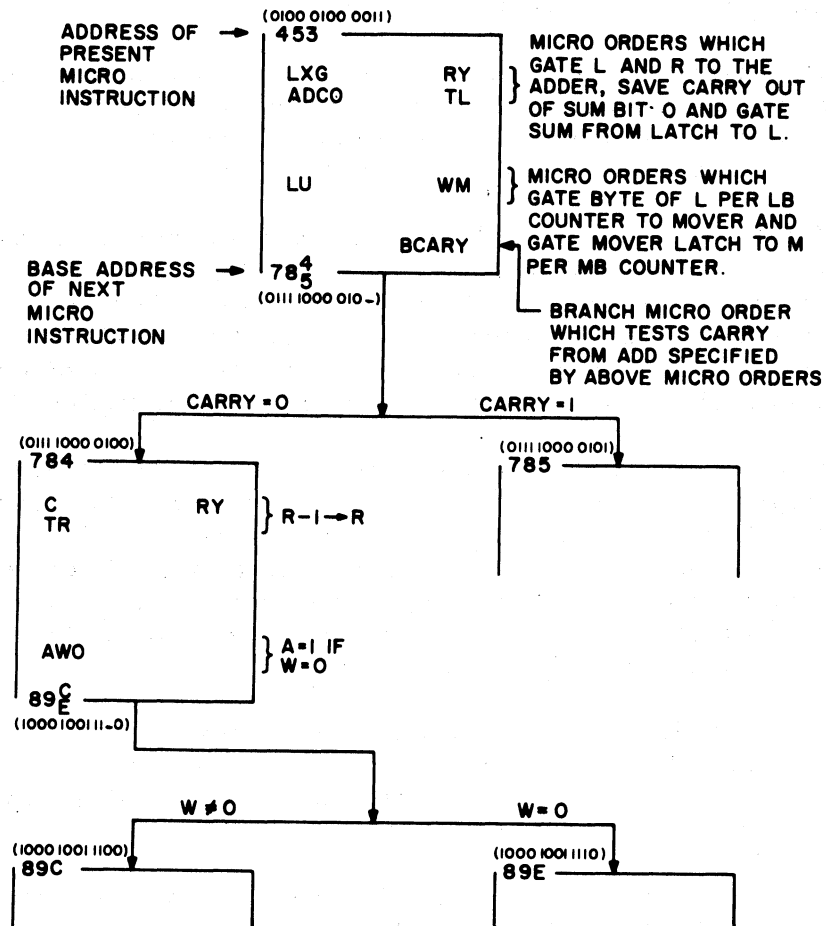


Figure 8. Flow chart representation of a microprogram.

required to decode the branch test and to form the ROS word address in time for selection of the next word.

The remaining thirty branch micro-orders test conditions in the immediately preceding cycle or conditions stored in stats, registers or counters.

The next microinstruction selected may be one of four by specifying another branch test forming the A-branch bit. There are 62 A-branch micro-orders in the Model 50.

The microprogram example in Figure 8 shows an A branch in ROS word 784. Branch micro-order AWO causes the A branch bit to be a one if the data in the mover latches, W, was zero in the preceding cycle.

One branch-order may test a logical combination of conditions such as either one of two counters being zero or one counter value being greater than another. Considerations made in the specification of branch micro-orders are discussed in the section, "Specification of Branch Test Micro-Orders."

Presumably the branching power of a ROS-controlled machine could be extended by the addition of more branch bits. For each bit, however, the ROS word must contain a control field for branch test micro-orders. The Model 50 has a 6-bit, A-branch test field and a 5-bit, B-branch test field. See Figure 7.

The use of function branching is another technique for increasing the branching capability. On the Model 50, ROS address bits 6-9 may be obtained directly from the present ROS word or from data within the CPU such as a hexadecimal multiplier digit. The address formed is then a function of the multiplier value; it is used to address a table of microinstructions, each controlling the fetching of a multiplicand multiple in local storage for the corresponding value of the multiplier digit.

Another example of function branching on the Model 50 is found in the instruction preparation microprogram. The 8-bit System/360 operation code is initially loaded into bits 0-7 of the M register. A function branch is taken on M bits, 0-3, determining the class of instruction, e. g., floating-point long, VFL character, etc. A second function branch is taken on remaining op-code bits 4-7 to address the first microinstruction of a routine. This routine executes a specific instruction within an instruction class such as VFL logical AND.

An address control bit in the ROS word determines whether bits 6-9 of the ROS address are to be obtained by a function branch or from the low-order base field. See Figure 7. If this field is not to be used directly for addressing, it is decoded to determine the source of the function branch.

A function branch may be used in conjunction with an A or a B-branch. Thus a branch on 2, 4, 16, 32, and 64 conditions is possible.

SPECIFICATION OF BRANCH TEST MICRO-ORDERS

The branching conditions should be kept general for maximum usage in all basic microprograms and future microprograms for special features. The GP stats allow the microprogrammer considerable branching flexibility. Each stat can have a totally different function per microprogram and sometimes even within the same microprogram, the stat having a succession of functions. On the Model 50, micro-orders for testing each one of the eight GP stats are included in both the A and the B branch fields.

Micro-orders are also assigned to test the most common branch conditions such as the adder latch = 0, the mover latch = 0, the adder carry, counters = 0, a counter value odd, etc. These are assigned to the A and B fields rather arbitrarily except where two branch conditions are known to be frequently used together, such as the adder carry and the adder latch = 0. These are assigned to different fields.

While the simple branch tests are given high usage, some special branch tests were required on the Model 50 to handle the floating-point and the VFL instructions. By judicious selection of these branches a significant increase in performance and a saving of ROS words was realized.

The handling of VFL character instructions will serve as an illustration. These instructions are unique in that :

1. both operands are contained in main storage;
2. each operand may start on any byte boundary;
3. the operands may be as long as 256 bytes and contained in as many as sixty-five main storage words.

These instructions are handled by fetching the first words of operands one and two to the M and L registers respectively. See Figure 1. Byte counters, MB and LB, controlling the selection of bytes from M and L to the mover are initially set to the starting address of each operand. One microinstruction selects byte pairs, operates on them, and stores the resultant byte to M. It is desirable to know in the same cycle when either byte selected is the last available in the CPU for that operand and when the last byte pair is being handled. Thus an A branch condition is chosen which tests $(G = 0 \text{ or } MB = 3)$ and a B branch condition which tests $(LB = 3)$. See Figure 9.

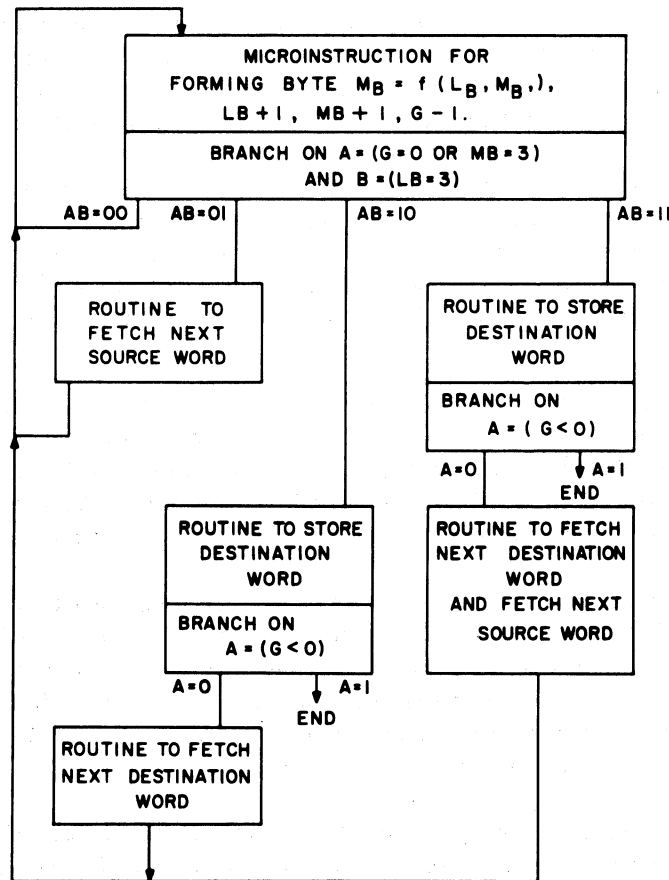


Figure 9. General program for byte handling of VFL character instructions.

CONCLUSIONS

There are many factors to be considered in the design of a read-only storage controlled processor. Among them are the realizable speed, width and depth of the ROS, the amount of data flow parallelism, the performance objectives of individual instructions, the amount of supporting control hardware permissible, and the ease of the addition of new microprograms.

ROS controls are used on the System/360 Models 20, 30, 40, 50, and 65. The fastest ROS is used on the Model 65 (having a 200-nsec cycle time) and has an access time (See Figure 4 for definition) of 70 nanoseconds using 10-nsec logic. There is no ROS presently available to match the 60-nsec cycle time of the Model 90 series which is one of the reasons that conventional hardware controls are used.

The width of a ROS increases with the number of independent data flow paths. The Model 30, having an 8-bit wide data flow, uses a 60-bit wide ROS; whereas, the Model 50, which has a 32-bit wide adder path, an 8-bit mover, and a local storage (as well as a main storage), has a 90-bit wide ROS word.

The depth of the ROS increases with the size and complexity of an instruction set. For a given instruction set it is difficult to express the number of words required in terms of other factors.

It can generally be said that the processors with multiple data paths tend to require fewer words. On the Model 50, addressing of main storage and byte handling in the mover can be controlled by the same word. On a smaller machine having one data path at least two cycles or words would be required.

The addition of some hardware in the data path can reduce the number of words required. In the Model 50 the use of hardware counters to select byte pairs for processing in the mover greatly reduced the number of words that would otherwise have been required. In the Model 40, which has a 2-byte wide data flow, separate programs are written for the handling of even and odd byte pairs. See Figure 10.

A particular algorithm used to execute an instruction can greatly affect the number of words used. For instance, the use of multiples of the multiplicand and a function branch on the multiplier digit in the floating point long instruction execution, as mentioned in the "Read-Only Storage Word Addressing" section, substantially increased performance at the cost of ROS words.

The Model 75 and Model 90 series have a high degree of simultaneity. That is, many independent data paths operate in parallel with the state of all data path controls not easily related on any one machine cycle. It is conceivable that as ROS technology improves, future processors may use several ROS units for controlling the independent control areas with synchronization and control between units provided by common states and branch conditions.

In determining the amount of supporting control hardware that is acceptable one must weigh the benefits derived from additional hardware; such as, increased performance or a reduction in ROS word width or depth against the cost, packaging, and maintainability of that hardware. The addition of the mover function register

in the Model 50 decreased the number of words because six instructions are executed by one microprogram. The use of special branches, such as the VFL branch as mentioned in the "Specification of Branch Test Micro-Orders" section, reduced the words and increased performance. The purist may consider the addition of such control hardware to supplement the ROS word controls as a deviation from basic ROS control philosophy. However, the objective is not to design an academically interesting machine but instead to produce a high performance, low-cost processor justifying some compromises to achieve this goal.

The attainment of the best control design depends upon the objectives and many restrictions of a particular project. Control design is still an art limited by the creativity of the designer.

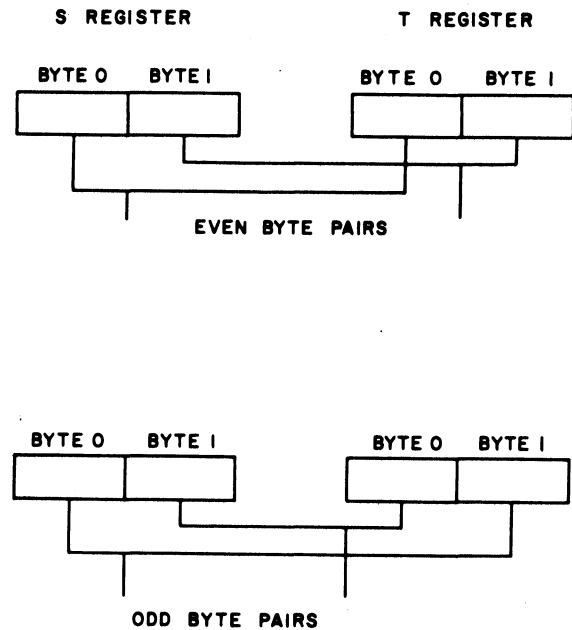


Figure 10. Even and odd byte pairs.

REFERENCES

1. G.A. Blaauw, F. P. Brooks, Jr. "The Structure of System/360" Part I -- "Outline of Logical Switches," IBM Systems Journal, Vol. 3, #213.
W. Y. Stevens, "The Structure of System/360" Part II -- "System Implementations," IBM Systems Journal, Vol. 3, #213.
2. L. J. Boland, "Analysis of Read-Only Memory for Control," Thesis for M.E.E. degree at Syracuse University, June 1963, IBM TIC 63E6-4362.